

10

9:45-9:55

$t = 5$

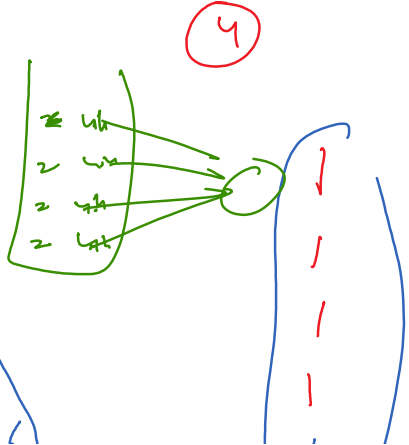
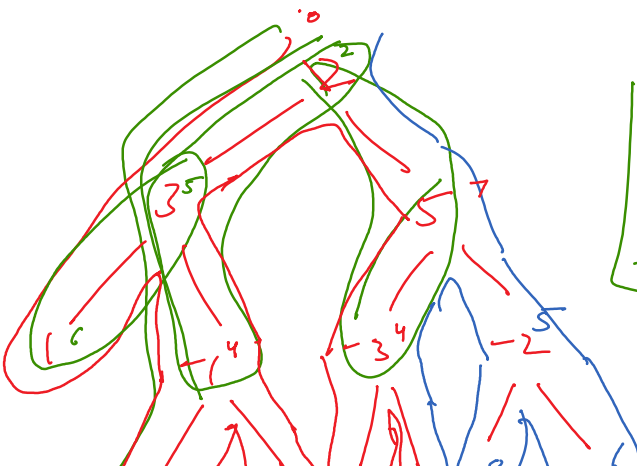
\downarrow ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓
 $0^1 \quad 2^2 \quad -1^1 \quad 0^1 \quad 6^7 \quad 3^{10} \quad -4^6 \quad -1^5 \quad 1^6 \quad 2^8 \quad 4^{12}$

$12 \rightarrow 1$
 $8 \rightarrow 1$
 $5 \rightarrow 1$
 $0 \rightarrow 1$
 $2 \rightarrow 1$
 $1 \rightarrow 12$
 $7 \rightarrow 1$
 $10 \rightarrow 1$
 $6 \rightarrow 12$

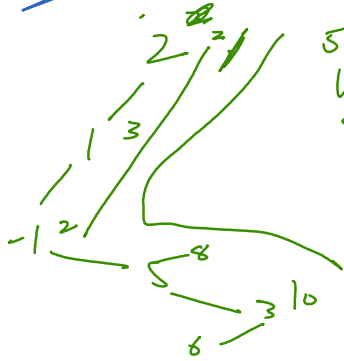
$1 + 2 + 1 + 2 + 1$

-1
 0
 6

$0 \rightarrow 1$
 ~~$2 \rightarrow 1$~~
 ~~$5 \rightarrow 12$~~
 ~~$1 \rightarrow 1$~~
 ~~$7 \rightarrow 1$~~
 ~~$4 \rightarrow 1$~~
 ~~$5 \rightarrow 1$~~
 ~~$6 \rightarrow 1$~~



~~6~~
~~5~~
~~8~~
~~6~~



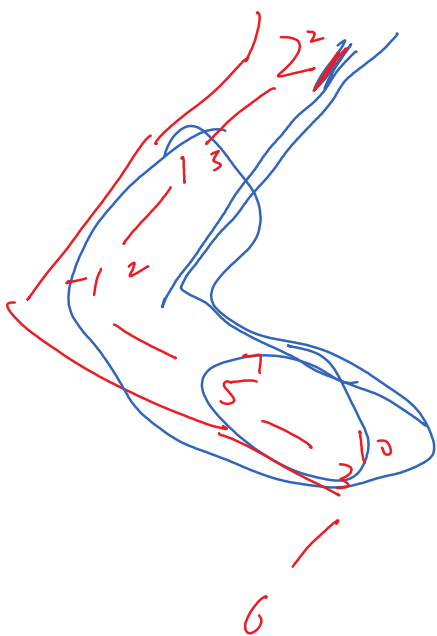
0 - 1
 2 - 1
 5 - 1
 6 - 1
 8 - 1

2 → 2



8

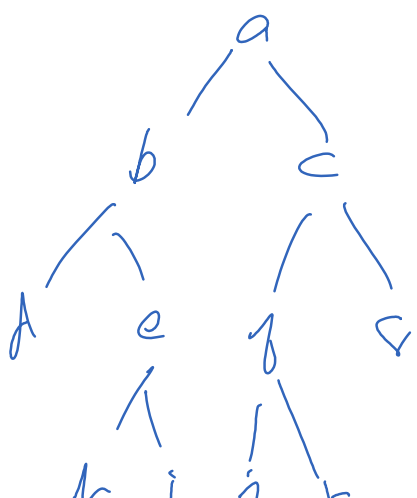
0 - 1
 2 - 1 → 1
 7 - 1
 10 - 1



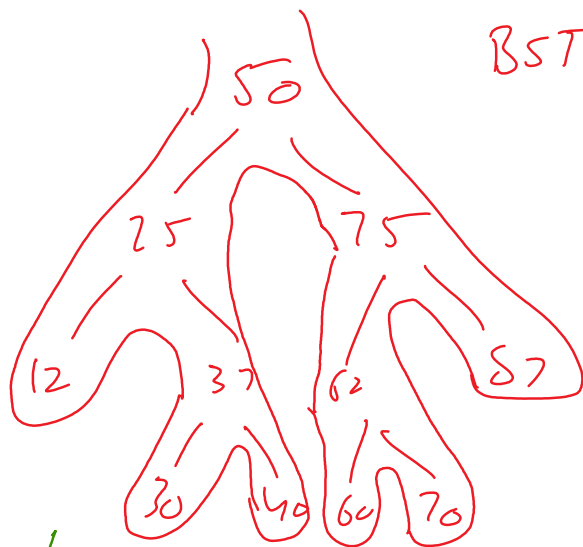
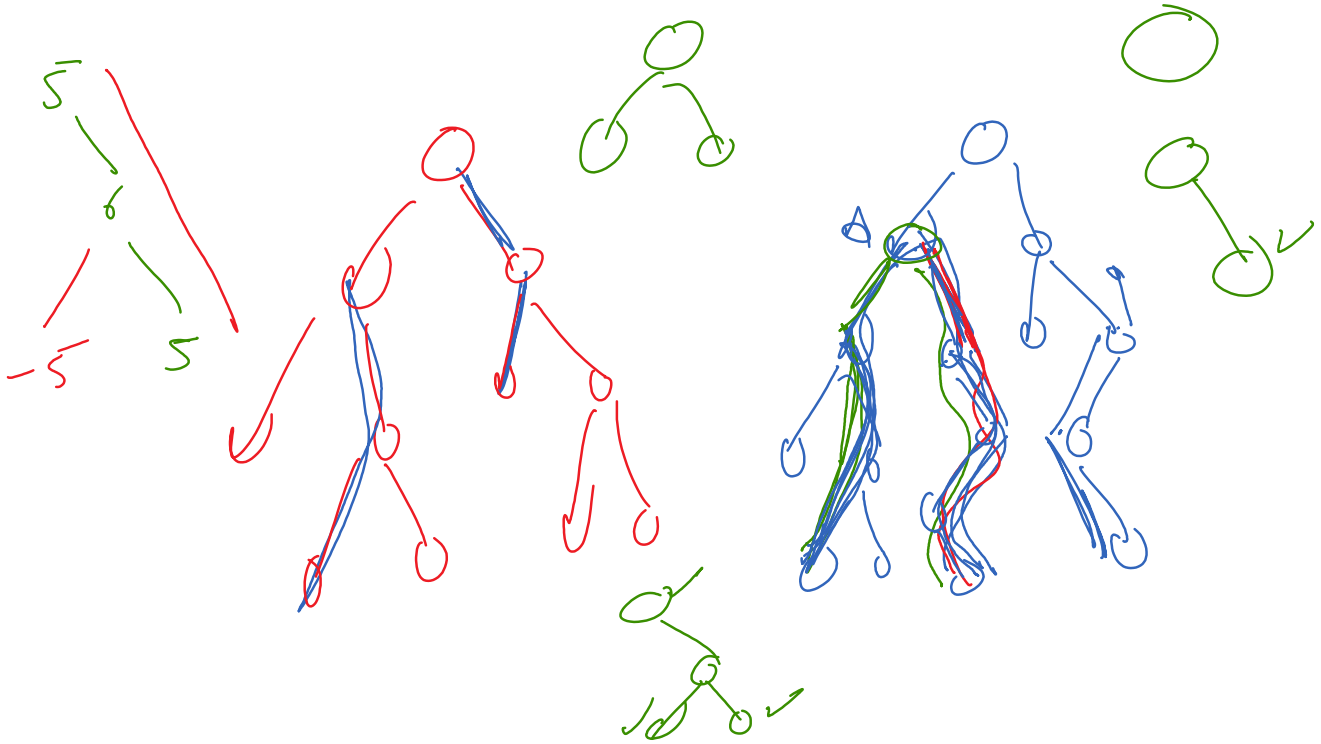
8

2

0 - 1
 2 - 2
 3 - 1
 7 - 1
 10 - 1



max sum from leaf to leaf
 10:31 to 10:41
 5



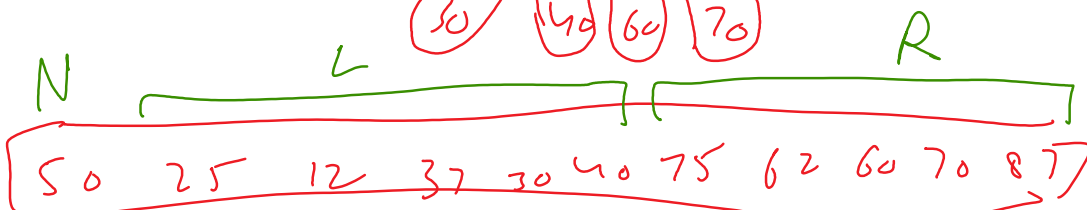
BST

1. Pre ✓

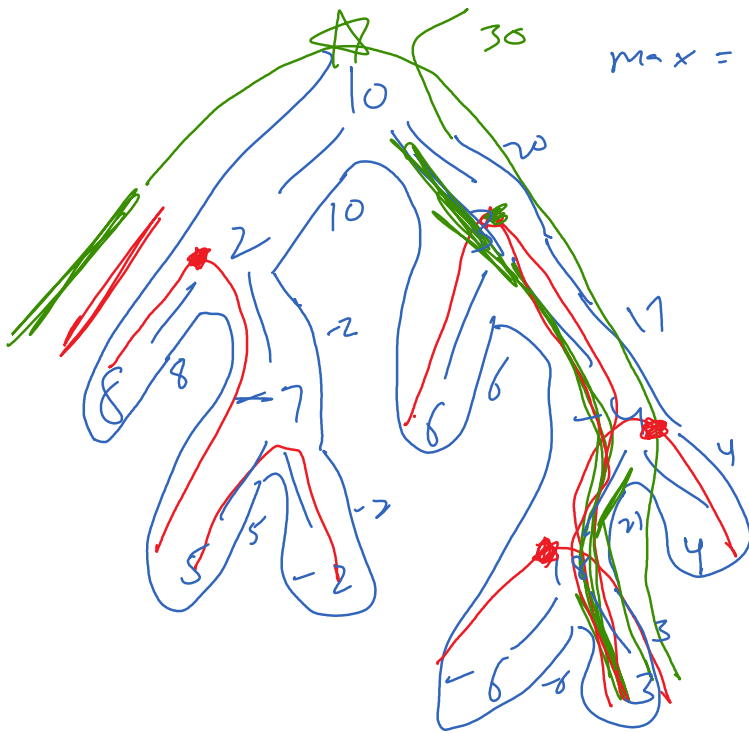
2. Pre-copy

sort

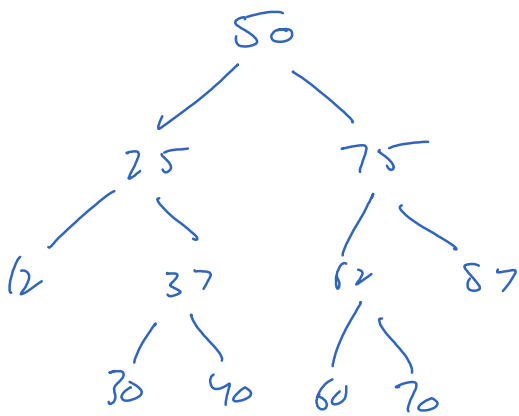
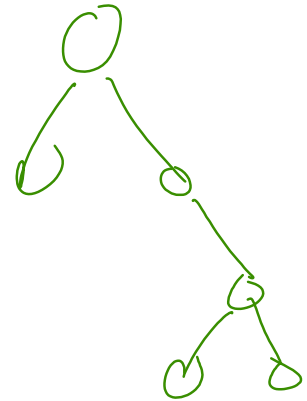
$(\Theta n)^2$



50 25 12 37 30 40 75 62 60 70 87



max = ~~-10~~ ~~14~~ 8 15 24
26
40

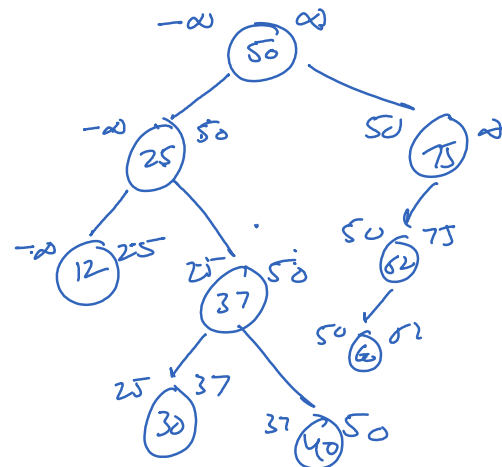


11:31 - 11:41

→ 50] N

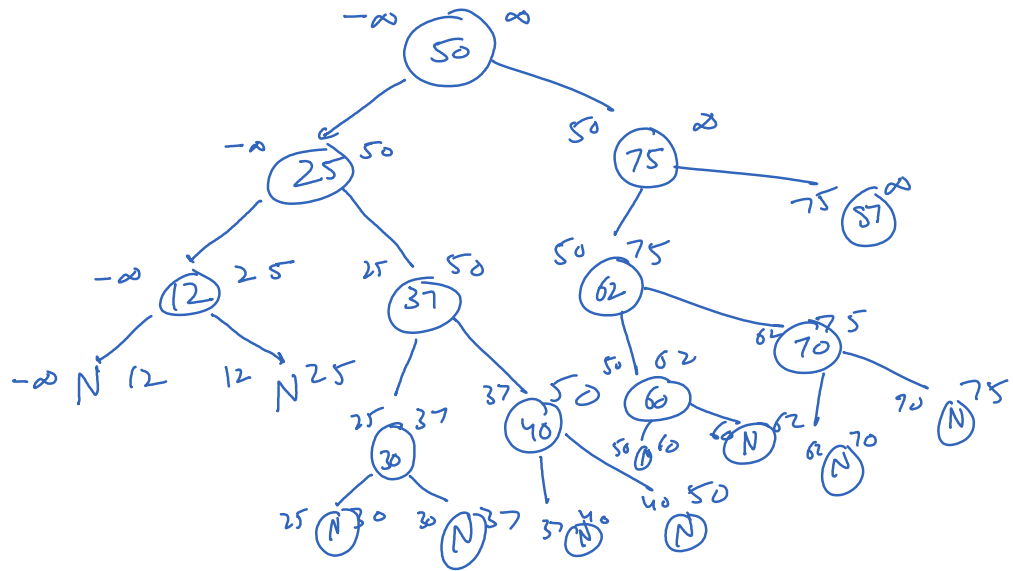
→ 25
→ 12
→ 37
→ 30
→ 40

→ 75
→ 62
→ 60
→ 70
87



Pre
Post
or
Optimal
Range

→ 50 → 25 → 12 → 37 → 30 → 40 → 75 → 62 → 60 → 70 → 87 →

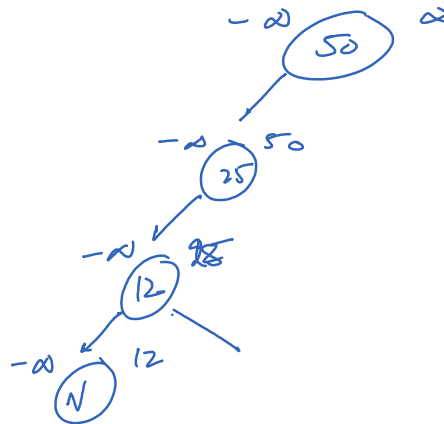


```
public TreeNode construct(int[] pre, int min, int max){
    if(index == pre.length){
        return null;
    } else if(pre[index] > min && pre[index] < max){
        TreeNode node = new TreeNode();

        node.val = pre[index];
        index++;

        node.left = construct(pre, min, node.val);
        node.right = construct(pre, node.val, max);

        return node;
    } else {
        return null;
    }
}
```

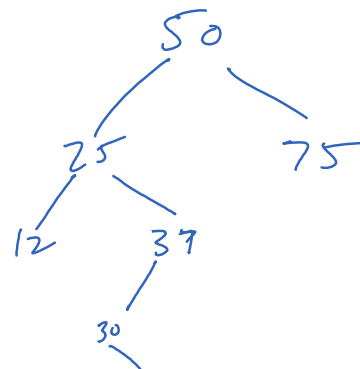


```
public TreeNode construct(int[] pre, int min, int max){
    if(index == pre.length){
        return null;
    } else if(pre[index] > min && pre[index] < max){
        TreeNode node = new TreeNode();

        node.val = pre[index];
        index++;

        node.left = construct(pre, min, node.val);
        node.right = construct(pre, node.val, max);

        return node;
    } else {
        return null;
    }
}
```

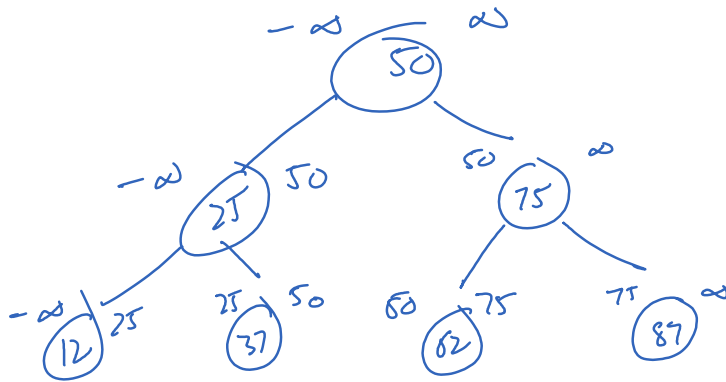


```

    return node;
  } else {
    return null;
  }
}

```

12 37 25 62 87 75 50



50₀ 25₁ 12₂ 37₃ 75₄ 62₅ 87₆

4	3	3	4	6	6	-1
---	---	---	---	---	---	----

